|  |  |
| --- | --- |
| Gerb-BMSTU_01 | **Министерство науки и высшего образования Российской Федерации**  **Федеральное государственное бюджетное образовательное учреждение**  **высшего образования**  **«Московский государственный технический университет**  **имени Н.Э. Баумана**  **(национальный исследовательский университет)»**  **(МГТУ им. Н.Э. Баумана)** |

ФАКУЛЬТЕТ \_\_\_***Робототехника и комплексная автоматизация***\_\_\_\_\_\_

КАФЕДРА \_***Системы автоматизированного проектирования (РК-6)***\_

**ОТЧЕТ ПО ПРОИЗВОДСТВЕННОЙ ПРАКТИКЕ**

Студент\_\_\_\_\_\_\_\_\_\_Гассиев Валерий Германович \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

*фамилия, имя, отчество*

Группа\_\_***РК6-81Б***\_\_\_

Тип практики \_\_\_\_\_\_\_\_\_\_\_***Преддипломная***\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Название предприятия\_\_\_\_\_\_\_\_***ИФК Тренинг***\_\_\_\_\_\_\_

Студент **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Гассиев В.Г.**

*подпись, дата фамилия, и.о.*

Руководитель практики

от кафедры **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Витюков Ф.А.**

*подпись, дата фамилия, и.о.*

Оценка \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

***2024 г.***

**«Московский государственный технический университет имени Н.Э. Баумана**

**(национальный исследовательский университет)»**

**(МГТУ им. Н.Э. Баумана)**

УТВЕРЖДАЮ

Заведующий кафедрой *РК6*

\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_ *А.П. Карпенко* \_

« \_\_\_ » \_\_\_\_\_\_\_\_\_\_\_\_ 2024 г.

**ЗАДАНИЕ**

**на прохождение производственной практики**

**\_\_\_\_\_\_*Преддипломная*\_\_\_\_\_\_**

Тип практики

Студент

\_\_\_\_\_\_Гассиев Валерий Германович\_\_\_\_\_\_\_\_\_ \_\_***4***\_\_ курса группы \_***РК6-81Б***\_

Фамилия Имя Отчество № курса индекс группы

в период с \_***13 мая 2024***\_\_\_ г. по ***26 мая 2024*** г.

*Предприятие:* \_\_\_\_\_\_\_\_\_***ИФК Тренинг***\_\_\_\_\_\_\_\_\_\_

*Подразделение:* \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(отдел/сектор/цех)

*Руководитель практики от предприятия (наставник):*

***Витюков Федор Андреевич***

(Фамилия Имя Отчество полностью, должность)

*Руководитель практики от кафедры:*

***Витюков Федор Андреевич***

(Фамилия Имя Отчество полностью, должность)

*Задание:*

**1.** *Провести анализ анимаций в веб-разрбаотке*

**2.** *Реализовать анимации в двух каруселях с помощью svelte/transition*

Дата выдачи задания ***14 мая 2024***г.

Руководитель практики от предприятия  **\_\_\_\_\_\_\_\_\_\_ /\_\_*Ф.А. Витюков*\_\_/**

Руководитель практики от кафедры  **\_\_\_\_\_\_\_\_\_\_ /\_\_*Ф.А. Витюков*\_/**

Студент **\_\_\_\_\_\_\_\_\_\_ /\_\_*В.Г. Гассиев*\_/**

СОДЕРЖАНИЕ

[ВВЕДЕНИЕ 5](#_Toc167389363)

[1. Теория анимаций в веб-разработке 6](#_Toc167389364)

[2. Основные принципы анимаций 7](#_Toc167389365)

[3. Виды анимации 8](#_Toc167389366)

[4. Технические аспекты реализации анимаций 10](#_Toc167389374)

[5. Реализация анимаций с помощью svelte/transitions 10](#_Toc167389387)

[6. Реализации анимации для переключения слайдов карусели с использованием svelte/transitions 13](#_Toc167389388)

[7. Реализации анимации текста и логотипа 15](#_Toc167389389)

[8. Будущее развитие анимаций в веб-разработке 17](#_Toc167389390)

[ЗАКЛЮЧЕНИЕ 19](#_Toc167389391)

[СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ 20](#_Toc167389395)

ВВЕДЕНИЕ

Анимации являются важной частью современного веб-дизайна и разработки, добавляя интерактивность и улучшая пользовательский опыт. В условиях жесткой конкуренции на рынке цифровых продуктов, создание привлекательного и интуитивно понятного интерфейса становится ключевым фактором успеха. Анимации играют в этом процессе важную роль, помогая решать множество задач, таких как улучшение визуального восприятия, управление вниманием пользователя и повышение общей эстетической привлекательности приложения.

Целью данного проекта является разработка веб-карусели с использованием Svelte и Tailwind, с особым акцентом на анимации переходов, реализованных с помощью библиотеки svelte/transitions. Карусель будет поддерживать как изображения, так и видео, предоставляя пользователям плавный и привлекательный интерфейс для навигации.

1. Теория анимаций в веб-разработке

Анимации в веб-разработке играют важную роль в создании привлекательных и интуитивно понятных интерфейсов. Они помогают пользователям лучше взаимодействовать с веб-приложениями, улучшая визуальное восприятие и общий пользовательский опыт. В этом разделе мы рассмотрим теоретические аспекты анимаций в веб-разработке, их значение, основные принципы, виды, технические аспекты реализации, а также примеры и лучшие практики.

**Значение анимаций в веб-разработке**

Анимации выполняют несколько ключевых функций в веб-разработке:

1. **Улучшение пользовательского интерфейса (UI**):

* Анимации делают интерфейс более динамичным и интересным.
* Плавные переходы и эффекты делают взаимодействие с веб-страницами более приятным.

1. **Обеспечение обратной связи**:

* Анимации могут указывать на результат действий пользователя (например, нажатие кнопки).
* Визуальные эффекты помогают пользователю понять, что его действие зарегистрировано.

1. **Привлечение и управление вниманием**:

* Анимации помогают направить внимание пользователя на важные элементы интерфейса.
* Они могут использоваться для акцентирования внимания на новых или важных частях страницы.

1. **Объяснение и информирование**:

* Анимации могут быть использованы для объяснения сложных процессов или навигации по веб-приложению.
* Они могут помочь пользователям понять, как использовать интерфейс.

1. Основные принципы анимаций

Создание эффективных анимаций требует соблюдения ряда принципов, которые помогают сделать анимации полезными и ненавязчивыми.

1. Цель

Каждая анимация должна иметь четкую цель и способствовать улучшению UX. Бессмысленные или чрезмерные анимации могут отвлекать пользователя и ухудшать восприятие интерфейса.

1. Естественность

Анимации должны быть плавными и естественными, чтобы не отвлекать пользователя. Они должны имитировать физические процессы и законы, такие как инерция и ускорение, чтобы казаться более реалистичными.

1. Производительность

Важно учитывать производительность при создании анимаций. Анимации не должны замедлять работу приложения или вызывать задержки. Необходимо использовать оптимизированные методы и технологии для их реализации.

1. Согласованность

Анимации должны быть согласованными по всему приложению, чтобы создать единый и связный пользовательский опыт. Это включает в себя использование одних и тех же типов анимаций для схожих действий и переходов.

1. Умеренность

Следует избегатьчрезмерного использования анимаций, чтобы не перегружать пользователя. Анимации должны быть использованы умеренно и только там, где они действительно необходимы.

1. Виды анимации

В веб-разработке существует несколько типов анимаций, которые используются для различных целей. Рассмотрим основные из них.

Переходы (Transitions)

Переходы применяются для плавного изменения свойств элементов, таких как цвет, размер, прозрачность и положение. Они помогают создать визуальные связи между состояниями элементов и делают изменения менее резкими.

**.button** {

**background-color**: blue;

transition: **background-color** 0.3s ease-in-out;

}

**.button**:hover {

**background-color**: green;

}

Листинг 1. Пример CSS-транзиции

**Трансформации (Transformations)**

Трансформации используются для изменения формы и положения элементов на странице. Они могут включать в себя масштабирование, вращение, сдвиг и наклон.

**.box** {

transform: scale(1);

transition: transform 0.3s ease-in-out;

}

**.box**:hover {

transform: scale(1.2);

}

Листинг 2. Пример CSS-трансформации

**Ключевые кадры (Keyframes)**

Ключевые кадры позволяют создавать сложные анимации, состоящие из нескольких этапов. Они позволяют определить промежуточные состояния элемента и создать плавные переходы между ними.

**@keyframes** **slideIn** {

**from** {

transform: translateX(-100%);

}

**to** {

transform: translateX(0);

}

}

**.element** {

animation: slideIn 0.5s ease-in-out;

}

Листинг 3. Пример CSS-анимации с ключевыми кадрами

Анимации на основе JavaScript

Анимации на основе JavaScript позволяют создавать более сложные и интерактивные эффекты, которые трудно реализовать с помощью CSS. JavaScript предоставляет возможность динамически изменять свойства элементов и управлять анимацией в реальном времени.

**const** element = document.querySelector('.element');

**function** animateElement() {

element.style.transition = 'transform 0.5s ease-in-out';

element.style.transform = 'translateX(100px)';

}

element.addEventListener('click', animateElement);

Листинг 4. Пример анимации на основе JavaScript

1. Технические аспекты реализации анимаций
2. Использование CSS для анимаций

CSS-транзиции и анимации являются наиболее простым и производительным способом реализации анимаций. Они могут быть аппаратно ускорены и не требуют использования JavaScript.

Преимущества использования CSS:

* Простота и лаконичность кода
* Аппаратное ускорение
* Легкая поддержка и кроссбраузерная совместимость

1. Использование JavaScript для анимаций

JavaScript предоставляет больше возможностей для создания сложных и интерактивных анимаций. Он позволяет управлять анимациями в реальном времени, реагировать на события и изменять свойства элементов динамически.

Преимущества использования JavaScript:

* Высокая гибкость и контроль
* Возможность создания интерактивных и сложных анимаций
* Поддержка анимаций на основе физики и пользовательских взаимодействий

1. Реализация анимаций с помощью svelte/transitions

Но в рамках проекта будем использовать библиотеку svelte/transitions для реализации анимаций. Svelte/transitions предоставляет мощные и удобные инструменты для создания анимаций, которые делают интерфейс более привлекательным и интерактивным. Это обусловлено, тем что поскольку проект разработан с использованием Svelte, svelte/transitions идеально интегрируется с фреймворком. Это позволяет легко добавлять анимации к компонентам, минимизируя количество необходимого кода. Так же, анимации, реализованные с помощью svelte/transitions, оптимизированы для работы в браузере и могут использовать аппаратное ускорение. Это обеспечивает плавные и эффективные анимации, не перегружая процессор. И наконец, Svelte/transitions позволяет настраивать параметры анимаций, такие как продолжительность, задержка и функция ускорения. Это дает возможность создавать уникальные эффекты, соответствующие требованиям дизайна и функциональности проекта.

Svelte/transitions - это библиотека для создания анимаций и переходов в приложениях, разработанных с использованием фреймворка Svelte. Она предоставляет простой и эффективный способ добавления анимаций к различным элементам интерфейса, таким как появление, исчезновение или изменение их состояния.

**Встроенные переходы**

Библиотека svelte/transitions предоставляет несколько встроенных переходов, которые можно использовать для создания различных анимаций. Некоторые из наиболее распространенных встроенных переходов включают в себя:

* **fade**: Плавное появление или исчезновение элемента с изменением прозрачности.
* **fly**: Анимация перемещения элемента по дуге или прямой линии с определенной скоростью и направлением.
* **slide**: Сдвиг элемента в указанном направлении с плавным появлением или исчезновением.
* **scale**: Изменение размера элемента с плавным увеличением или уменьшением его размера.

**Настройка параметров переходов**

Каждый встроенный переход может быть настроен с помощью различных параметров, таких как продолжительность анимации, задержка перед началом анимации, функция ускорения (easing function) и дополнительные эффекты.

Примеры параметров, которые можно настроить для каждого перехода, включают в себя:

* **duration**: Продолжительность анимации в миллисекундах.
* **delay**: Задержка перед началом анимации.
* **easing**: Функция ускорения, определяющая, как изменяется скорость анимации со временем (например, "ease", "ease-in", "ease-out", "ease-in-out" и другие).
* **opacity**: Начальное и конечное значение прозрачности элемента.

**Примеры использования встроенных переходов**

Рассмотрим пример использования встроенных переходов для создания анимации появления и исчезновения элементов:

**<script>**

**import** { fade } from 'svelte/transition';

**let** visible = **false**;

**</script>**

**<button** on:click={() =**>** visible = !visible}>

Toggle

**</button>**

{#if visible}

**<div** transition:fade**>**

This element fades in and out

**</div>**

{/if}

Листинг 5. Использование встроенных переходов

**Создание кастомных переходов**

Кроме встроенных переходов, вы также можете создавать собственные кастомные переходы с помощью библиотеки svelte/transitions. Это позволяет полностью контролировать поведение анимации и создавать уникальные эффекты, соответствующие вашим потребностям и дизайну интерфейса.

Процесс создания кастомных переходов включает в себя определение начальных и конечных стилей, а также логику анимации с помощью CSS и JavaScript.

1. Реализации анимации для переключения слайдов карусели с использованием svelte/transitions

В данном разделе рассмотрим пример реализации переключения слайдов карусели с использованием библиотеки svelte/transitions. Это позволит создать плавные анимации переходов между изображениями и видео, улучшая пользовательский опыт.

**<div** class="p-0 relative w-full overflow-hidden h-[80vh] xsm:h-[20vh] sm:h-[40vh] md:h-[45vh] lg:h-[55vh] xl:h-[70vh] 2k:h-[85vh] 4k:h-[82vh] 8k:h-[82vh]"**>**

{#if carouselItems && carouselItems.length > 0}

{#if mainImageIndex === carouselItems.length - 1 && carouselItems[mainImageIndex].video}

**<div** class="video-container w-full h-[80vh] xsm:h-[20vh] sm:h-[40vh] md:h-[45vh] lg:h-[55vh] xl:h-[70vh] 2k:h-[85vh] 4k:h-[82vh] 8k:h-[82vh]"**>**

**<iframe**

title="video"

src={getYouTubeEmbedUrl(carouselItems[mainImageIndex].video)}

allow="autoplay; encrypted-media"

allowfullscreen

on:timeupdate={handleVideoTimeUpdate}

on:playing={startTimer}

in:fly="{{ x: direction \* 1000, duration: 500, easing: cubicOut }}"

out:fly="{{ x: -direction \* 1000, duration: 500, easing: cubicOut }}"

**></iframe>**

**</div>**

{:else}

{#key mainImageIndex}

**<img** class="w-full h-full " alt={`Image ${mainImageIndex + 1}`} src={carouselItems[mainImageIndex].image}

in:fly="{{ x: direction \* 1000, duration: 250, easing: cubicOut }}"

out:fly="{{ x: -direction \* 1000, duration: 250, easing: cubicOut }}" **/>**

{/key}

{/if}

{/if}

**</div>**

Листинг 6. Код карусели

**Контейнер карусели:**

Контейнер для карусели (div) имеет класс relative w-full overflow-hidden, который делает его контейнером для перекрывающихся элементов. Высота контейнера задается с помощью классов Tailwind CSS, что позволяет адаптировать его размер под разные разрешения экранов.

**Проверка наличия элементов в карусели:**

Используется конструкция {#if carouselItems && carouselItems.length > 0}, чтобы убедиться, что в карусели есть элементы для отображения. Если элементы присутствуют, они будут отображаться, иначе блок не будет выводиться.

**Отображение видео:**

Внутри конструкции {#if mainImageIndex === carouselItems.length - 1 && carouselItems[mainImageIndex].video} проверяется, является ли текущий элемент последним в массиве и содержит ли он видео. Если условие выполняется, отображается видео с помощью тега <iframe>.

В iframe задается URL видео, а также атрибуты allow и allowfullscreen, чтобы разрешить воспроизведение и полноэкранный режим.

Для анимации переходов используется встроенный переход fly из библиотеки svelte/transitions. Параметры перехода (x: direction \* 1000, duration: 500, easing: cubicOut) задают начальную и конечную позиции элемента, продолжительность анимации и функцию ускорения.

**Отображение изображений:**

Если текущий элемент не является видео, отображается изображение с помощью тега <img>. Для уникальной анимации каждого изображения используется блок {#key mainImageIndex}, чтобы Svelte мог следить за изменениями индекса.

Картинка имеет класс w-full h-full, чтобы занимать всю высоту и ширину контейнера.

Для анимации переходов также используется переход fly, но с другой продолжительностью (duration: 250). Это обеспечивает плавное появление и исчезновение изображений при смене слайдов.

1. Реализации анимации текста и логотипа с использованием svelte/transitions

В данном примере мы рассмотрим реализацию анимации текста и логотипа в карусели с использованием svelte/transitions.

**<div** class="absolute top-0 left-0 w-full h-full"**>**

{#each carouselItems as { text, position, logo, platforms }, index (text)}

{#if index === mainImageIndex}

**<div**

style={`top: ${position.top}; left: ${position.left}; bottom: ${position.bottom};`}

class="text-white 8k:text-[250px] 4k:text-[140px] 2k:text-[64px] xl:text-5xl xl:text-[43px] lg:text-[33.68px] md:text-[26px] sm:text-xl xsm:text-[9px] font-extrabold 8k:leading-[270px] 4k:leading-[147px] 2k:leading-[77px] lg:leading-[39px] leading-tight absolute ${position.classPos} sm:mt-[-3%] md:mt-[-3%] lg:mt-[-3%] xl:mt-[-3%] 2k:mt-[-3%] 4k:mt-[-3%] 8k:mt-[-3%]"

transition:fly|local={{ x: direction \* 200, duration: 800, easing: cubicOut }}

**>**

{#if shouldCenterText(index)}

**<div** class="text-center"**>**

{@html text.split('**<br/>**').join('**<br** **/>**')}

**</div>**

{:else}

{@html text.split('**<br/>**').join('**<br** **/>**')}

{/if}

**</div>**

{/if}

{/each}

{#each carouselItems as { logo }, index (logo)}

{#if index === mainImageIndex && logo}

**<div**

style={`top: ${logo.position.top}; left: ${logo.position.left};`}

class="absolute"

transition:fly|local={{ x: direction \* 200, duration: 800, easing: cubicOut }}

**>**

**<img**

src={logo.image}

alt="Logo"

class={logo.sizeClasses}

**/>**

**</div>**

{/if}

{/each}

**</div>**

Листинг 7. Код для анимации текста и логотипа слайда

**Контейнер компонента:**

Контейнер для анимаций (div) имеет класс absolute top-0 left-0 w-full h-full, что делает его абсолютным контейнером, занимающим всю ширину и высоту родительского элемента. Это позволяет свободно размещать элементы внутри контейнера.

**Анимация текста:**

Используется конструкция {#each carouselItems as { text, position, logo, platforms }, index (text)} для итерации по элементам карусели. Внутри каждой итерации проверяется, соответствует ли текущий индекс mainImageIndex, чтобы отобразить только активный элемент.

Текстовый блок позиционируется с помощью встроенного стиля style={top: ${position.top}; left: ${position.left}; bottom: ${position.bottom};} и класса absolute, что позволяет точно определить его местоположение на экране.

Текст стилизуется с использованием классов Tailwind CSS, таких как text-white, font-extrabold и адаптивных классов для различных разрешений экранов.

Для анимации перехода текста используется встроенный переход fly с локальными параметрами (transition:fly|local={{ x: direction \* 200, duration: 800, easing: cubicOut }}). Параметры перехода задают начальную и конечную позиции элемента, продолжительность анимации и функцию ускорения.

**Центрирование текста:**

Проверяется условие shouldCenterText(index), чтобы определить, должен ли текст быть центрированным. Если условие выполняется, текст оборачивается в блок с классом text-center.

**Анимация логотипа:**

Внутри конструкции {#each carouselItems as { logo }, index (logo)} проверяется, соответствует ли текущий индекс mainImageIndex, и есть ли логотип у текущего элемента.

Логотип позиционируется с помощью встроенного стиля style={top: ${logo.position.top}; left: ${logo.position.left};} и класса absolute.

Логотип отображается с использованием тега <img>, с заданием пути к изображению и классов для его размера (class={logo.sizeClasses}).

Для анимации перехода логотипа также используется переход fly с локальными параметрами.

1. Будущее развитие анимаций в веб-разработке

Веб-анимации играют ключевую роль в создании динамичного и привлекательного пользовательского опыта. С постоянным развитием технологий и веб-стандартов ожидается, что будущее анимаций в веб-разработке будет направлено на более качественное, ресурсоэффективное и интерактивное взаимодействие. Рассмотрим несколько тенденций, которые могут определить будущее веб-анимаций:

**Векторные анимации и SVG:**

Векторные анимации, основанные на SVG (масштабируемой векторной графике), становятся все более популярными благодаря своей масштабируемости и простоте анимации. Они позволяют создавать плавные и адаптивные анимации, что особенно важно для мобильных устройств.

**WebGL и 3D-анимации:**

С появлением WebGL и развитием графических движков в веб-браузерах, веб-разработчики получили доступ к созданию высококачественных 3D-анимаций и визуализаций. В будущем это может привести к расширенному использованию 3D-графики для создания интерактивных веб-приложений и игр.

**Анимации на основе алгоритмов и искусственного интеллекта:**

Применение алгоритмов машинного обучения и искусственного интеллекта может изменить способ, которым анимации создаются и управляются. Автоматизированные инструменты могут помочь веб-разработчикам создавать более интуитивные и динамичные анимации на основе данных и поведения пользователей.

**Микроанимации для повышения UX:**

Микроанимации, такие как анимированные иконки, кнопки или переходы, становятся неотъемлемой частью веб-дизайна для улучшения пользовательского опыта. В будущем, с развитием технологий, можно ожидать более интегрированных и проработанных микроанимаций для создания плавных и интуитивных пользовательских интерфейсов.

**Производительность и оптимизация:**

С увеличением количества анимаций на веб-страницах важно обеспечивать их высокую производительность и оптимизацию. В будущем, разработчики будут активно использовать инструменты для управления ресурсами, такие как CSS-анимации, а также инструменты профилирования и оптимизации для обеспечения плавной работы даже на устройствах с ограниченными ресурсами.

ЗАКЛЮЧЕНИЕ

Использование Svelte и его модуля Transition в веб-разработке представляет собой эффективный способ создания динамичных и привлекательных анимаций. Модуль Transition обеспечивает простой и интуитивно понятный подход к созданию анимаций, позволяя разработчикам сосредоточиться на сути анимации, а не на технических деталях её реализации.

За счет использования Svelte, анимации становятся частью компонентного подхода к веб-разработке, что обеспечивает чистоту кода, легкость сопровождения и повторное использование. Кроме того, Svelte позволяет оптимизировать производительность анимаций за счет минимизации объема генерируемого кода.

Будущее анимаций в веб-разработке с использованием Svelte и Transition обещает еще более широкое применение динамичных эффектов, более гармоничное взаимодействие с пользователем и более высокую производительность веб-приложений. Основанные на инновационных подходах к созданию анимаций, проекты, использующие Svelte и Transition, будут на передовой в обеспечении удовлетворительного пользовательского опыта и современного веб-дизайна.

СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ

1. Svelte Documentation // Svelte Documentation URL: <https://svelte.dev/docs>. Дата обращения: [10.05.2024];
2. CSS Tricks // CSS Tricks URL: <https://css-tricks.com/>. Дата обращения: [18.10.2023];
3. MDN Web Docs // MDN Web Docs URL: <https://developer.mozilla.org/en-US/>. Дата обращения: [11.05.2024];
4. W3Schools // W3Schools URL: <https://www.w3schools.com/>. Дата обращения: [20.05.2024];
5. Transition - Svelte // Svelte Documentation URL: <https://svelte.dev/docs#svelte_transition>. Дата обращения: [20.05.2023]